Module 1 – Table Expressions

Overview

The main purpose of this lab is to familiarise yourself with key features of table expressions with TSQL.

- Views
- In-line table valued functions
- Derived tables
- Common table expressions
- Temporary tables

Objectives

At the end of this lab, you will be able to:

- Create a new view in SQL
- Create a new in-line table valued function with and without parameters in SQL
- Use a derived table in SQL
- Create a common table expression
- Create temporary tables and check their scope

Setup: Launch SQL Server Management Studio

1. On the Start menu, click All Programs, click Microsoft SQL Server 2014 and then click SQL Server Management Studio.
2. The Microsoft SQL Server Management Studio window opens, and then the Connect to Server dialog box will appear.
3. In the Connect to Server dialog box, click Connect to accept the default settings.
4. On the toolbar, click New Query, and either select the QATSQLPLUS database in the Available Databases box, or type USE QATSQLPLUS in the query window.
5. If you wish, you may save your queries to My Documents or the desktop. All modules are separate and you will not require any queries from this module in any later module.

Exercise 1: Views

In this exercise, you will create a view to return all the courses available with the vendor name.

The main tasks for this exercise are as follows:

1. Create a query that returns the correct information.
2. Convert the query to a view called CourseList.
3. Use the CourseList view.

Task 1: Create the query

1. Write a query to join the Course and Vendor tables. The column shared by the tables is VendorID. The columns returned should be:
   - CourseName
   - CourseID
   - VendorName
2. Test the query.
3. Keep the query window open for the following tasks.

Task 2: Create a view

1. Using the query from Task 1, create a view called dbo.CourseList.
2. Test the view dbo.CourseList using a SELECT query.
3. Keep the query window open for the following tasks.

Exercise 2: In-line Table-Valued Function (TVF)

In this exercise, you will create two inline TVF to return the number of courses attended and number of days training received per delegate. The first should return the information for all delegates, and the second should return the information for a specific DelegateID.

The main tasks for this exercise are as follows:

1. Create a query that returns the correct information.
2. Convert the query to a TVF called udf_DelegateDays. Test the function udf_DelegateDays.
3. Create a second TVF called udf_IndividualDelegateDays to use the same basic query but only return the result for a specific DelegateID.

Task 1: Create the query

1. Write a query to join the Delegate, DelegateAttendance and CourseRun tables. The columns returned should be:
   - DelegateID
   - Sum(DurationDays) → DelegateDays
   - Count(*) → DelegateCourses
2. Test the query.
3. Keep the query window open for the following tasks.

Task 2: Create a TVF

1. Using the query from task 1, create a TVF called udf_DelegateDays. As this will be an in-line table valued function the return will be type TABLE. No parameters are to be passed to this function.
2. Test the TVF using a SELECT query.
3. Keep the query window open for the following tasks.

Task 3: Create a TVF

1. Using the query from task 1, create a TVF called udf_IndividualDelegateDays. As this will be an in-line table valued function the return will be type TABLE. A parameter called @DelegateID (data type INT) should be passed to this function.
2. Test the TVF using a SELECT query passing the DelegateID 1.
3. Keep the query window open for the following tasks.

Exercise 3: Derived tables

In this exercise, you will write a query that returns all delegates that have been taught by Jason Bourne.

The main tasks for this exercise are as follows:

1. Create a query that returns the CourseRunID and StartDate for each course taught by Jason Bourne.
2. Incorporate the query from task 1 as a derived table and return the delegate information.

Task 1 : Create the derived table query

1. Write a query to join the Trainer and CourseRun tables. The only columns returned should be the CourseRunID and StartDate where the trainer name was Jason Bourne.
2. Test the query and ensure 2 rows are returned.
3. Keep the query window open for the following tasks.

Task 2 : Create the full query

1. Write a query to join the DelegateAttendance, Delegate and derived table from task 1. The columns returned from the query should be:
    - DelegateID
    - DelegateName
    - CompanyName
    - StartDate
2. Test the query and ensure 19 rows are returned.
3. Keep the query window open for the following tasks.

Exercise 4: Common Table Expression (CTE)

In this exercise, you will write a query that returns all delegates that have been taught by Jason Bourne by using a CTE rather than a derived table.

The main tasks for this exercise are as follows:

1. Create a query that returns the CourseRunID and StartDate for each course taught by Jason Bourne.
2. Incorporate that query inside a CTE.
3. Use the CTE result in a join and return the delegate information.

Task 1 : Create the initial query

1. Write a query to join the Trainer and CourseRun tables. The only columns returned should be the CourseRunID and StartDate, where the trainer name was Jason Bourne. (Reuse the query from the previous Exercise, Task 1).
2. Test the query and ensure 2 rows are returned.
3. Keep the query window open for the following tasks.

Task 2 : Create the CTE

1. Take the query from the previous task and write a CTE called BourneCourses.
2. Test the query and ensure 2 rows are returned.
3. Keep the query window open for the following tasks.

Task 3 : Change the query to join to other tables

1. Take the CTE query from the previous task and join it to the Delegate and DelegateAttendance tables. The columns returned from the query should be:
   - DelegateID
   - DelegateName
   - CompanyName
   - StartDate
2. Test the query and ensure 19 rows are returned.
3. Keep the query window open for the following tasks.

Exercise 5: Temporary Tables

In this exercise, you will write a query that returns and stores all Microsoft courses into temporary tables and view the scopes of the temporary table types.

The main tasks for this exercise are as follows:

1. Create a query that returns all Microsoft courses.
2. Create and insert the data into both a local and global temporary table.
3. Check the availability of the temporary tables in a different session.

Task 1: Create the Microsoft course query

1. Write a query to select all columns from the dbo.Course table where the VendorID = 2.
2. Run the query and ensure 6 rows are returned.
3. Keep the query window open for the following tasks.

Task 2: Create and use temporary tables

1. Using the query from task 1, select the query result into #MicrosoftLocal and ##MicrosoftGlobal. The SELECT INTO code takes the form:

   ```
   SELECT <columns>
       INTO <destination table name>
       FROM <source table(s)>
   ```

2. In the same query window, write a query to view the content of both #MicrosoftLocal and ##MicrosoftGlobal.
3. Run the queries and ensure 6 rows are returned from each.
4. Keep the query window open for the following tasks.

Task 3: Review availability of the temporary table to other sessions

1. On the toolbar, click New Query.
2. In the new query window, write a query to view the content of both #MicrosoftLocal and ##MicrosoftGlobal.
3. Run the queries separately. The global temporary table (##MicrosoftGlobal) should return 6 rows. The local temporary table will return an error.
4. Keep the query window open for the following tasks.

Answers

The answers below are for example only. Coding style and order of columns should not matter.

Ex 1 Task 1

```
SELECT V.VendorName, C.CourseName, C.CourseID
   FROM dbo.Vendor AS V
      INNER JOIN dbo.Course AS C
        ON V.VendorID = C.VendorID
```

Ex 1 Task 2

```
CREATE VIEW dbo.CourseList AS
   SELECT V.VendorName, C.CourseName, C.CourseID
      FROM dbo.Vendor AS V
         INNER JOIN dbo.Course AS C
           ON V.VendorID = C.VendorID
GO
SELECT * FROM dbo.CourseList
```

Ex 2 Task 1

```
SELECT D.DelegateID,
   SUM(CR.DurationDays) AS DelegateDays,
   COUNT(*) AS DelegateCourses
   FROM dbo.Delegate AS D
      INNER JOIN dbo.DelegateAttendance AS DA
        ON D.DelegateID = DA.DelegateID
      INNER JOIN dbo.CourseRun AS CR
        ON CR.CourseRunID = DA.CourseRunID
   GROUP BY D.DelegateID
```

Ex 2 Task 2

```
CREATE FUNCTION udf_DelegateDays()
   RETURNS TABLE
   AS
   RETURN (
      SELECT D.DelegateID, SUM(CR.DurationDays) AS DelegateDays,
            COUNT(*) AS DelegateCourses
         FROM dbo.Delegate AS D
            INNER JOIN dbo.DelegateAttendance AS DA
               ON D.DelegateID = DA.DelegateID
            INNER JOIN dbo.CourseRun AS CR
               ON CR.CourseRunID = DA.CourseRunID
         GROUP BY D.DelegateID
   )
GO
SELECT * FROM dbo.udf_DelegateDays()
```

Ex 2 Task 3

```
CREATE FUNCTION dbo.udf_IndividualDelegateDays(@DelegateID INT)
   RETURNS TABLE
   AS
   RETURN(
      SELECT @DelegateID AS DelegateID,
             SUM(CR.DurationDays) AS DelegateDays,
             COUNT(*) AS DelegateCourses
         FROM dbo.Delegate AS D
             INNER JOIN dbo.DelegateAttendance AS DA
                ON D.DelegateID = DA.DelegateID
             INNER JOIN dbo.CourseRun AS CR
                ON CR.CourseRunID = DA.CourseRunID
         WHERE D.DelegateID = @DelegateID
   )
GO
SELECT * FROM dbo.udf_IndividualDelegateDays(1)
```

Ex 3 Task 1

```
SELECT CourseRunID, StartDate
   FROM dbo.Trainer AS T
      INNER JOIN dbo.CourseRun AS CR
         ON T.TrainerID = CR.TrainerID
   WHERE TrainerName = 'Jason Bourne'
```

Ex 3 Task 2

```
SELECT D.DelegateID, D.DelegateName, D.CompanyName, JB.StartDate
   FROM dbo.Delegate AS D
      INNER JOIN dbo.DelegateAttendance AS DA
         ON D.DelegateID = DA.DelegateID
      INNER JOIN (
         SELECT CourseRunID, StartDate
            FROM dbo.Trainer AS T
               INNER JOIN dbo.CourseRun AS CR
                  ON T.TrainerID = CR.TrainerID
            WHERE TrainerName = 'Jason Bourne'
         ) AS JB
         ON DA.CourseRunID = JB.CourseRunID
```

Ex 4 Task 1

```
SELECT CourseRunID, StartDate
   FROM dbo.Trainer AS T
      INNER JOIN dbo.CourseRun AS CR
         ON T.TrainerID = CR.TrainerID
   WHERE TrainerName = 'Jason Bourne'
```

Ex 4 Task 2

```
WITH BourneCourses AS (
   SELECT CourseRunID, StartDate
      FROM dbo.Trainer AS T
         INNER JOIN dbo.CourseRun AS CR
            ON T.TrainerID = CR.TrainerID
      WHERE TrainerName = 'Jason Bourne'
)
SELECT *
   FROM BourneCourses
```

Ex 4 Task 3

```
WITH BourneCourses AS (
   SELECT CourseRunID, StartDate
      FROM dbo.Trainer AS T
         INNER JOIN dbo.CourseRun AS CR
            ON T.TrainerID = CR.TrainerID
      WHERE TrainerName = 'Jason Bourne'
)
SELECT D.DelegateID, D.DelegateName, D.CompanyName, JB.StartDate
   FROM dbo.Delegate AS D
      INNER JOIN dbo.DelegateAttendance AS DA
         ON D.DelegateID = DA.DelegateID
      INNER JOIN BourneCourses AS JB
         ON DA.CourseRunID = JB.CourseRunID
```

Ex 5 Task 1

```
SELECT *
   FROM dbo.Course
   WHERE VendorID = 2
```

Ex 5 Task 2

```
SELECT * INTO #MicrosoftLocal
   FROM dbo.Course   WHERE VendorID = 2
SELECT * INTO ##MicrosoftGlobal
   FROM dbo.Course   WHERE VendorID = 2
GO
SELECT *
   FROM #MicrosoftLocal

SELECT *
   FROM ##MicrosoftGlobal
```